自動チューニング : 数理的手法によるソフトウェア高性能化 (次世代計算科学の基盤技術とその展開)

AUTHOR(S):

須田, 礼仁

CITATION:

須田, 礼仁. 自動チューニング : 数理的手法によるソフトウェア高性能化 (次世代計算科学の基盤技術とその展開). 数理解析研究所講究録 2013, 1848: 1-14

ISSUE DATE:

2013-08

URL:

http://hdl.handle.net/2433/195093

数理解析研究所講究録
第1848巻 2013年 1-14

自動チューニング：数理的手法によるソフトウェア高性能化

須田礼仁
東京大学情報理工学系研究科
Reiji Suda
Graduate School of Information Science and Technology,
the University of Tokyo

1 ソフトウェア性能チューニング

ソフトウェアの機能を保ったままで，性能を改善させるようなプログラムの変更作業をチューニングと呼ぶ．ここでいう**性能**は，所要時間のようなコストで表現されることが多い．時間のほか，電力，エネルギー（電力効率），数値誤差などがコストとなりうる．また，プログラムの**変更**は，ループアンローリングのようなプログラム変換，データ構造の設計のほか，クイックソートの再帰の深さのようなパラメタの設定も含む．このような実装の違いを**変種**と呼び，その選択を表すパラメタを**チューニングパラメタ**という．

我々が取り組んできたチューニングの事例には，以下のようなものがある．

- **行列積** [1]: 2つの密行列どうしの積を計算するルーチン．我々はABCLibScript [2] によって自動的に生成されるルーチンを用いている．すなわち，単純な行列積ルーチンに指示行を加えてABCLibScriptプリプロセッサを通すと，指定された段数に**アンロール**された行列積ルーチンが生成される．異なる段数にアンロールされたルーチンの中で，所要時間が短いもの，あるいは消費電力が少ないものを選択したい．

- **疎行列ベクトル積** [3]: 疎行列と密ベクトルの積を計算するルーチンは，Krylov部分空間法などの反復法で基本的な構成要素である．疎行列を表現するデータ構造は「フォーマット」と呼ばれ，COO, CRS, BRS, JDS, DIAなど多様な手法が知られている．フォーマットの違いにより，計算量，メモリ使用量，メモリアクセスの順序等が異なる．どのフォーマットがどのような性能を出すかは，格納する疎行列の非零パターン（非零要素の位置）に依存する．与えられた行列に対して，行列ベクトル積の所要時間ができるだけ短いフォーマットを選択したい．

- **代数的多重格子法（AMG）** [4]: AMGは疎行列を係数行列とする大規模連立一次方程式を解くアルゴリズムで，工学院大学藤井昭宏講師が開発しているライブラリを使わせていただいている．AMGのアルゴリズムは多様なパラメタを有している．たとえば，粗い行列を決めるパラメタ，多

重の格子間を行き来するサイクル，スムーザーに用いられる緩和法の種類とパラメタなどである．どのようなパラメタが高性能を与えるかは，係数行列および右辺ベクトルにより異なる．

チューニングにより高性能を達成するには，個々の条件に合った適切なチューニングパラメタの選択をしなければならない．ここで**条件**とは，コストに影響を与える任意の要因で，レジスタ数やキャッシュサイズ，コンパイラの最適化，行列サイズや非零パターン，行列・ベクトルの要素の値など，多様な要因が含まれる．従来の手作業によるチューニングは，ハードウェア，ソフトウェア，データを適当に与えて性能測定を繰り返し，およそ良好な性能を示すパラメタを選択するというものであった．

自動チューニングは，このチューニングの作業の自動化を目指す．期待される効果はおよそ2つある．第1に，自動化により，人手で行うよりも徹底した性能の比較と最適化ができ，より高い性能を達成することである．すなわち，人的コストの削減である．第2に，さまざまな条件の下で自動的にチューニングを実施することにより，多様な条件下でより高い性能を達成することである．すなわち，条件の多様性に対する適応能力をソフトウェアに付与することである．

自動チューニングを実現するためには，いくつか重要な研究領域がある．第1は，チューニングのための変種をどのようにプログラミングするか，といった問題を扱う，プログラミング言語の領域．第2は，チューニングパラメタの制御，所要時間や電力などのコストの測定や，情報収集・蓄積などを扱う，システムソフトウェアの領域．第3は，性能測定のための実験計画や，コストとチューニングパラメタの関係を分析し，最適なチューニングパラメタを探す，数理の領域．第4は，並列計算機やGPU，あるいは行列計算やアプリケーションなどで実際に有効な変種を開発する，個別チューニング手法の領域．

この節では，著者が取り組んできた事例を挙げつつ，自動チューニングを実現する手法を解説する．

1.1 チューニング可能なソフトウェアの実装

まず，前節で挙げたようなチューニングパラメタを持つソフトウェアを実装する．これにはいくつかの要素が含まれる．まず，高性能を導くようなプログラムの変種を実装することである．これは個々の計算（行列積，疎行列ベクトル積，線形ソルバ等）に依存する．コンパイラの最適化が提供するようなより汎用性の高いものも存在する．条件によって高性能であったり，逆に低性能になったりするような変種も考えられる．

変種には4つの種類がある．第1に，**スケジューリング変種**は，計算内容は同一であるが，その順序が異なるものである．二重ループの外側ループと内側ループを交換するなどのループ変換や，命令列の順序，並列処理におけるスケジューリングなどが含まれる．また，計算結果が複数回参照される場合に，結果を保存しておくか，同じ計算を再度行うか，という違いも広い意味でのスケジューリングである．さらには，条件によってAまたはBを計算するというときに，あらかじめAとBとを両方計算しておいて，条件によって計算結果を選択するのも，広い意味でのスケジューリングである．第2に，**アルゴリズム変種**は，同一の処理を行う複数のアルゴリズムのひとつを選択するものである．例えばソートのような基本アルゴリズムや，連立一次方程式の解法には複数のものがある．常微分方程式の解法や，偏微分方程式の離散化手法（差分法，有限要素法，スペクトル法，粒子法）なども広い意味でのアルゴリズム変種と言える．第3に，**データ構造変種**は，同一の論理的情報を格納するための複数のデータ構造のひとつを選択するものである．密行列における列優先・行優先の違いや，3次元格子を3次元配列に入れる際にx, y, zの3つの次元をどの配列次元に配置するか，というのもデータ構造変種である．複素

数の配列として，(1) 複素数型を定義してその配列とする，(2)2 倍の長さの実数の配列を準備して，偶数番要素に実部を，奇数番要素に虚部を格納する，(3) 実部用と虚部用に 2 つの実数の配列を準備する，といった違いがある．また，疎行列用の各種の格納形式，複素数の極形式など，さらには，ファイル圧縮に用いられる符号化によりデータを圧縮するのもデータ構造変種と言える．第 4 に，**プラットフォーム依存コーディング**がある．今般の CPU に広く採用されている短ベクトル（SIMD）演算のための特殊命令やデータ構造の使用，GPU のための CUDA，OpenCL，OpenACC といったプログラミング言語の使用，OpenMP や MPI などの並列処理がその例である．

次に，これらの変種を指定して実行できるように**チューニングパラメタ**を設ける．チューニングパラメタには，それを制御するためのインタフェースが伴わなければならないが，そのインタフェースの適切な選択は，チューニングが行われる**タイミング**（**コンパイル時**，**インストール時**，**実行時**）にもよる．すなわち，チューニングのための実行を発行する主体が，ビルドツールであるか，インストーラーであるか，アプリケーションであるかによって，必要とされるチューニングパラメタのインタフェースが異なる．例えば，ソフトウェア自身の内部に自動チューニング機構を組み込む場合には，ターゲットとなる手続きの引数の一つとして，あるいは大域変数として，チューニングパラメタを実装するなどの方法が考えられる．

さらに，時間や電力などの**コストを測定するルーチン**を実装し，そのインタフェースを提供する．また，行列のサイズや非零要素数など，チューニングの参考になるような情報（**特徴量**と呼ぶ）があれば，これを取得できるようにソフトウェアを実装し，インタフェースを提供する．特徴量以外の動的な条件は，コストに影響を与えるが自動チューニングで参照されないため，**擾乱要因**となる．

このほか，ソフトウェア，ハードウェア，開発者や利用者の都合により課される**制約条件**があれば，自動チューニング機構がそれを認識し，またそれに応じて適切に実行を制御できるようなインタフェースの実装が必要である．例えば，使用可能なメモリ量の上限などが考えられる．

実装において，ソフトウェア上で実現される，チューニングパラメタ，コスト，特徴量，制約条件などをあわせて**自動チューニングの 4 つの D** と呼んでいる．これらは自動チューニングに必要な要素のうち，ターゲットソフトウェアに実現され，自動チューニング機構に対してインタフェースが与えられるものである．自動チューニング機構は，与えられたインタフェースを通してこれらの情報を獲得し，パラメタを調整して，自動チューニング機能を実現する．

ところで，自動チューニングでは，データを与えて実際にハードウェアで実行することにより，チューニングパラメタとコストの関係を観測し，それに基づいてパラメタを最適化する．自動チューニングのためのソフトウェアの**実行**には，2 つの種類がある．ひとつは**実施**であり，これは実際のデータを与えて行われる実用的な計算であり，計算結果はアプリケーションで用いられる．このとき，チューニングパラメタを変更してコストを測定することは実際の計算においてオーバーヘッドとなりうることに注意が必要である．

もうひとつは**試行**であり，これは，コストを観測するために仮のデータを与えて行う実行である．このとき条件は実際の実行条件に一致しているとは限らないところに注意が必要である．また，試行のために**仮のデータおよびそれを供給する手段**を実装する必要がある．さらに，チューニングの対象がソフトウェア全体ではなく一部の場合には，**チューニング対象のみを切り出す**といった実装が行われることも多い．

これらにより，自動チューニングのターゲットとなるソフトウェアの準備が完了する．

1.2 事前知識と仮定の整理

1.2.1 コストモデル

チューニングパラメタ i，特徴量 j，擾乱 x の時のコスト c は数学的に

$$c = f(i, j, x)$$

と表すことができる．一般に，我々はこの関数 $f(i, j, x)$ を完全には知らない．しかし，ある種の知識を持っていることがある．

ハイパフォーマンスコンピューティングの分野では，以前から，チューニングパラメタや特徴量とコストとの関係を分析することが行われてきた．例えば行列積をアンローリングしたとき，行列サイズがアンローリング段数で割り切れればよいが，剰余が出る場合には余った計算を実行するための残余ループが必要となる．段数が大きいアンローリングループの性能が高いので，各行列サイズに対して，アンローリング段数が大きく剰余ループが小さくなるようなアンローリング段数が高性能になる [1]．

疎行列ベクトル積の場合，行列の非零パターンによって各フォーマットの実行性能が異なる [3]．非零パターンは規則的格子による偏微分方程式の求解のような予測可能なアプリケーションと，FEM や AMR のように実行時にならないとわからないアプリケーションとがある．後者の場合でも，Krylov 部分空間法などに用いられると，同じ行列に対してベクトルの積が数十回から数千回反復されるので，その間にチューニングをすることもできる．

AMG の場合，チューニングパラメタが性能に与える影響の大きさにある傾向が認められる [4]．まず，粗い行列をどのように定義するかにより，AMG の収束性が最も大きく影響される．次に，AMG をそれだけで解法とするのか，あるいは CG 法，BiCGSTAB 法，IDR 法等の前処理として用いるのかにより，収束性が大きく変わる．また，スムーザーやサイクルも性能に一定の影響を及ぼす．

また，コストについての性質も知られている．所要時間については，一定のばらつきが観測される．反復的な計算の場合，キャッシュミスが多い 1 回目の反復の所要時間が遅い．消費電力については，所要時間よりもばらつきが大きく，また時間粒度が粗い．温度が上昇するとリーク電流が増加するため，消費電力が増加する．このため，単に対象とする計算だけではなく，実行時の温度によって消費電力が決まる．

これらの知識から，チューニングパラメタ i，特徴量 j を与えた時のコスト c を

$$c \approx \tilde{f}(i, j)$$

のように推定する**コストモデル**を構築することができることがある．関数 $\tilde{f}$ には実験的に決めるモデルパラメタが含まれていてもよい．例えば各特徴量 j は実数 t_j に対応づけられていて，

$$\tilde{f}(i, j) = a_{i0} + a_{i1} t_j + a_{i2} t_j^2 + a_{i3} t_j^3$$

といったモデルが考えられる．ここで a_{ij} は未知パラメタで，観測データからフィッティングで求めることを想定している．

上記のモデルは，t_j に関しては 3 次多項式を想定しているが，i とコストに関して特定の傾向を仮定していないが，これでもモデルとして問題ない．例えば j とコストに関する 3 次多項式の近似をやめて

$$\tilde{f}(i, j) = a_{ij}$$

とするというのも，十分に意味のあるモデルである．

著者は多くの場合に**ベイズ的な性能モデル**を利用してきた．例えば上記の $\tilde{f}(i,j)=a_{ij}$ というモデルだけでは，「i,j とコストとの関係はどんなに極端であるかもわからない」ことを意味している．すなわち，ある (i,j) についてのコストを推定するためには，その (i,j) の組み合わせで少なくとも 1 回は観測することが必要である．しかしこれにベイズ的なモデル

$$a_{ij} \sim N(a_{00}, \tau^2)$$

を追加すると，a_{ij} に平均 a_{00} 分散 τ^2 の正規分布という事前情報を与えることができる．ここでパラメタ a_{00}, τ^2 は実験的に推定されるものでもよい（そのような推定方法として，階層ベイズ法や経験ベイズ法がある）．このようにすると，観測されていない (i,j) に対しても「平均的な」 a_{ij} が推定される．このため，例えば $a_{ij}=a_{00}-3\tau$ となるような a_{ij} が見つかれば，正規分布の性質から，その他の選択肢の約 99.9% は（観測されたことがあるかどうかによらず）a_{ij} 以上のコストを持つと期待される．すなわち，観測されたことのない組み合わせがあっても一定の判断をすることができるのである．複数のチューニングパラメタがあれば，選択肢の数は組合せ的に増加するため，すべての選択肢について観測するということが現実的でないような場面も十分想定されるから，観測されていない選択肢についてコストの推定ができることは重要である．

そのほかにも，特徴量が原因で観測ができないこともある．例えば我々が研究してきた電力に関する自動チューニングでは，前述のようにプロセッサの温度が重要な特徴量である．ところが，プロセッサの温度はソフトウェアから自由に制御することができない．とりわけ，低温は再現が難しい．長時間計算機をアイドルにし，空調の温度を低く風量を強く設定しなければ低温が達成できないうえ，一度計算が始まるとすぐに高温になってしまう．このため低温の状態で多数の実験を行うことは極めて困難である．従って，観測されていない選択肢が残っても何らかの最適化ができることが必要である．

1.2.2 擾乱モデルと未来の実行に関する仮定

擾乱要因は観測されないから，コストモデルに含めることはできない（擾乱を推定する場合は特徴量に含めるものとする）．従って，上記のコストモデルの近似が擾乱の下でどういう意味なのか明確にしておく必要がある．例えば擾乱が定常的であるとして，擾乱に対するコスト c の平均値を $\langle c \rangle$ で表現し，

$$\langle f(i,j,x) \rangle \approx \tilde{f}(i,j)$$

と設定することができる．

擾乱に関する仮定は必要である．例えば電力測定の場合，測定装置の物理的なゆらぎにより正規分布的な測定誤差が得られる．所要時間の測定の場合，プロセッサのクロックの進み具合によって測定するため，所要時間には処理内容によって決まる非負の下限がある．非対称行列向けのクリロフ部分空間反復法では，解法と問題の組み合わせによっては収束しないことがある．この最後の例のように，**条件によっては計算結果が得られない**ような場合には自動チューニングのためにそれに対応できるアルゴリズムを選ばなければならない．従って，この点について仮定を明らかにすることが重要である．モデルもそれに対応するものが必要で，例えば計算が（ある時間以内に）終了するかどうかを表す変数 $g(i,j,x) \in \{0,1\}$ を設けて，これに対するモデルを仮定するなどが考えられる．

例えば行列サイズによらず一定のアンローリング段数を選ぶとすると，行列サイズが擾乱要因の一つとなる．このとき平均値 $\langle c \rangle$ は，どのような行列サイズがどのような頻度で出現するかに関する仮定に基づいて定義されることになる．このように，性能モデルには，陽にあるいは暗に，**将来の実行における条件**（特徴量，擾乱要因）に関する仮定が含まれることがある．

以下の最適化問題としての定式化においても，未来のソフトウェア実行における条件に関する仮定が必要である．そもそもチューニングとは未来におけるソフトウェアの実行の効率化を目指すものであるので，ソフトウェアが未来にどのように実行されるかという仮定は必須なのである．

ソフトウェアが未来にどのように使われるかを仮定するには2つの方法がある．ひとつは**履歴ベース**の方法であり，過去の利用条件を外挿して未来の利用条件を推定する．もう一つは**計画ベース**の方法であり，ターゲットソフトウェアをどのような条件でどの程度利用するのか，計画として与える．

なお，**観測できる条件**を特徴量として取り出すか，擾乱要因として無視するかは重要な判断である．コストに大きな影響を与える要因を擾乱要因として処理してしまうと，擾乱の分散が大きくなり，測定結果が収束するまでにより多くの実験が必要となる．しかしそれを特徴量として処理するには，それがコストに与える影響を適切にモデル化する必要がある．さらに，モデルが複雑になれば，それだけ，モデルが適切な精度で構築されるまでの実験回数が必要となる．すなわち，自動チューニングの効率のためには，上手なモデルの選択が重要である．なお，**ランダムではない条件**をランダムな擾乱として扱うと適切なチューニングが達成されないので，特徴量として取り出すことが必須となる．

1.2.3 最適化問題としての定式化

自動チューニングを最適化問題として捉えると，目的関数と制約条件を明らかにする必要がある．自動チューニングの目的関数や制約条件は，自動チューニングのタイミングにもよる．

例えば，試行をまったく行わず，実施のみで自動チューニングする方法を**オンライン自動チューニング**と言う．ターゲットの計算が K 回実施されるとすると，オンライン自動チューニングの目的関数は，例えば K 回の実施の合計コスト

$$O_1 = \sum_K f(i_k, j_k, x_k)$$

とすることができる．ここで i_k, j_k, x_k 等は第 k 回の実行でのチューニングパラメタ，特徴量，擾乱要因である．チューニングパラメタを決定する関数

$$i_k = i(j_k, G_{k-1}, F)$$

を**戦略**という．ここで G_{k-1} は第1回〜第 $k-1$ 回までの実行で得られる情報，F は未来の実行に関する情報であり，戦略は特徴量 j_k，事前情報 G_{k-1}，与えられた未来の実行に関する情報 F に基づき決定することができる．

他方で，試行のみにより実施前に自動チューニングを完了する方式を**オフライン自動チューニング**と呼ぶ．試行を K 回行い，その結果，特徴量 j に対して最適と推定された選択肢を $i_{opt}(j)$ とすると，例えば

$$O_2 = \kappa^{-1} \sum_k f(i_k, j_k, x_k) + \langle\langle f(i_{opt}(j), j, x) \rangle\rangle$$

のように設定できる．ここで κ はユーザが定める定数であり，試行コストと実施コストとのバランスを決定する．ここで $\langle\langle f(i_{opt}(j),j,x)\rangle\rangle$ は未来の実行におけるコストの平均値であり，x および j に関する仮定された確率分布による平均値を表す．特に，**特徴量の分布の仮定**が必要である．また，試行回数 K は測定結果によって変化させうるので，戦略の一部となる．すなわち，いつ試行をやめるか（**停止時**）の問題を含む．なお，上記 O_2 の式で $\kappa^{-1} \to 0$ とすると，当然 $K \to \infty$ となる（試行が終わらない）ので，有限の κ を与えることが重要である．

このような，コストモデル，擾乱や将来の実行に関する仮定，目的関数などをあわせて**自動チューニングの 4 つの A** と呼んでいる．これらは自動チューニングを数理的に定式化するにあたって必要な仮定である．これらはターゲットソフトウェアに内在するものではない．ターゲットソフトウェアをどのように利用するか，あるいはどのように自動チューニングしたいか，というユーザの立場を表明するもの，あるいはコストモデルのような開発者が持っている事前知識を表現するものである．

1.3 自動チューニングのための数理的手法

自動チューニングの数理的手法は，上記の 4 つの A に基づき構成される．自動チューニングに必要な数理アルゴリズムには，測定データに関するデータ解析，コストモデルや擾乱モデルのパラメタフィッティング，次の試行・実施においてどのチューニングパラメタを選択するかという実験計画，オフライン自動チューニングにおいては停止時の決定と実施で用いる選択肢を選ぶ最適化などがある．このほか，現在の情報でどの程度最適化されていそうか定量的に評価できると便利である．これらの数理手法を**自動チューニングの 4 つの C** と呼んでいる．

なお，統計的な手法を用いる場合，ランダム性を確保するために，選択肢のインデクスをランダマイズすることも重要である．

2 事例：温度に依存した消費電力モデル

計算機の電力に関する最適化の必要性は，様々な意味で必要となってきている．第一に，高い消費電力は多くの熱を発生するので，それを冷却する必要がある．半導体の進歩により計算機のクロック周波数は速くなっているが，消費電力とチップ冷却能力の限界から，近年クロック周波数の伸びは極めて鈍化している．消費電力を制御することにより，プロセッサあたりの処理能力をより高く引き出せると期待される．第二に，高い消費電力は，それを供給できる電力設備および冷房設備を必要とする．スーパーコンピュータセンターの電力設備および冷房設備は，すべてのプロセッサを最大限に使用しても耐えられるように設計されているが，通常はその限界の半分程度で運用されている．ソフトウェア的に消費電力を制御できれば，設定された上限に見合う電力・冷房設備を設置すれば足りるし，逆に，提供できる電力・冷房設備よりも高い性能の計算機を設置することができる．第三に，高い消費電力は，エネルギー資源を多く消費し，電気代として経済的なコストもかかる．同じ計算が少ない消費電力で実現できれば，地球と予算にやさしい計算となる．

これらの要因から，プロセッサの消費電力をソフトウェア的にコントロールできるように研究が進められている．本稿執筆時点では，ソフトウェア的に制御でき，消費電力に影響を及ぼすハードウェアの「つまみ」は，クロック周波数・電源電圧制御（DVFS）など少数であるが，今後そのようなつまみが増

えてくることが期待されている．また，ハードウェアの消費電力を測定するには特殊な測定装置と接続する必要があるが，今後は実行時電力が容易に取得できるようになると期待される．

この節では，自動チューニングのための数理手法のひとつとして，電力に関する自動チューニングのための電力モデルを論ずる．ソフトウェアの具体的な実装や測定結果は別の機会に報告することとし，本稿では数理的なモデルで議論をする．

2.1 温度を考慮した消費電力モデル

本稿では，チューニングパラメタ i が与えられていて，選択肢 $I = \{1, 2, \ldots, M\}$ のうちひとつの値を取りうるものとする．また，プロセッサの温度 t が測定できるものとする．温度は離散化されて与えられ，N 通りの値 $T = \{t_1, t_2, \ldots, t_N\}$ を取るものとする．

温度を測るのは，計算機の消費電力が温度によって影響を受けるためである．プロセッサの消費電力は，ゲートのスイッチングにより消費される**動的電力**と，リーク電流などによる**静的電力**とがある．近年ではこの両者は同程度になっているが，温度が上昇するとリーク電流が増加し，静的電力が増加する．

このことは，自動チューニングにおいて無視できない影響を及ぼす．温度の効果を無視し，擾乱要因として扱うとすると，低温時に測定された選択肢は消費電力が少ないので，「低消費電力な選択肢」に見えてしまい，消費電力の推定を誤ってしまう．ランダムに選択肢を選びつつ，十分に長時間にわたって測定を繰り返せば，温度の効果は平均化されて擾乱要因とみなせるが，これは余計な擾乱要因を増やし，自動チューニングの効率を下げてしまう．そこで本稿では，温度の効果を考慮した消費電力モデルを作ることで，温度の効果を適切に考慮した効率的な自動チューニングを目指す．

これまでに自動チューニングでコストとしてよく取り上げられてきたのは所要時間である．所要時間に比べると，消費電力は測定精度が低く，物理的なばらつきの影響を大きく受ける．このため，温度による消費電力の効果は明白なものの，詳細なモデルを構築するのは現実的ではない（詳細なモデルを作成してもフィッティングするための情報を集めるのにコストがかかりすぎる）．そこで今回は，静的電力が温度に対して線形に増加すると想定して平均消費電力モデルを

$$\mu_{ij} \approx a_i + bt_j$$

とする．ここで μ_{ij} は選択肢 i, 温度 t_j での**平均消費電力**，a_i および b は未知定数である．より詳細に，ベイズモデル

$$\mu_{ij} = a_i + bt_j + \delta_{ij}, \quad \delta_{ij} \sim N(0, \tau^2)$$
$$y_{ijk} = \mu_{ij} + \epsilon_{ijk}, \quad \epsilon_{ijk} \sim N(0, \sigma^2)$$

と仮定する．ここで δ_{ij} はモデルと平均消費電力のずれ，y_{ijk} は選択肢 i，温度 t_j での k 回目の測定における**消費電力**，ϵ_{ijk} は測定ごとのばらつきであり，分散 σ^2, τ^2 は簡単のため既知の定数とする．さらに，未知定数に関するベイズモデルも

$$a_i \sim N(a_0, \tau_a^2), \qquad b \sim N(0, \tau_b^2)$$

を仮定する．ここでも平均 a_0 は未知定数，分散 τ_a^2 および τ_b^2 は既知の定数とする．

2.2 モデルに関する議論

ここで上記のモデルの選択について論ずる．

平均消費電力モデルは

$$\mu_{ij} \approx a_i + bt_j$$

であるから，温度に依存する項 bt_j は選択肢 i に依らない．すなわち消費電力を最小にすると期待される選択肢は，温度によらず a_i を最小にする i である．従って，温度ごとの平均消費電力 μ_{ij} ではなく基礎消費電力 a_i を推定し，選択肢 i を選ぶという方法もありそうである．

しかし実際には a_i を推定するという方法は自動チューニングには向かない．これは，温度 t_j が自由に制御できないことによる．温度はランダムな動きもあるが，消費電力の高い処理の後で上昇し，消費電力の低い処理の後で下降し，その傾向はランダムではない．特に前述のように低温での測定回数が限られるため，a_i を十分な精度で推定することが困難なのである．

これに対して，頻繁に観測される温度における情報は容易に集積するので，そのような温度に対する平均消費電力 μ_{ij} は適切に推定できる．また，測定されていない温度に対する消費電力も，モデルからある程度の精度で推定することができる．また，選択肢ごとに温度依存性が若干異なる場合にも適切に対応できる．

次に温度への影響である b の項については，選択肢によらず一定としている．選択肢によって温度への依存性が変化するより詳細なモデル，例えば $\mu_{ij} \approx a_i + b_i t_j$ は考えられるが，この場合には各 i について複数の温度で実行した結果がなければ b_i が決定できない．消費電力の測定ばらつきは大変大きく，そのばらつきの中から係数 b_i を選択肢 i ごとに決定するのは，あまり現実的ではないと思われる．このような場合には，およその一定の傾向を bt_j で与え，それからの誤差 δ_{ij} を許容する本モデルがより現実的と思われる．

未知定数に関するベイズモデル $a_i \sim N(a_0, \tau_a^2)$ は，未測定の選択肢に関する性能情報を与えることができる．また，一種の縮小推定が実現されるので，推定精度を向上させる効果も期待できる．

もうひとつのベイズモデル $b \sim N(0, \tau_b^2)$ も一定の効果が期待できる．上述のように，温度および消費電力の両方の測定のばらつきの大きさのゆえに，温度係数 b を正確に推定するには多数の実験データが必要である．実験数が少ない範囲では，b の推定が不安定になり，極端に大きい値や（非現実的な）負の値が出ることも少なくない．b に関するベイズモデルは，b の推定として極端な値を出す危険性を低減させ，推定を安定させる効果がある．もし消費電力に関する温度効果について事前情報があれば $b \sim N(b_0, \tau_b^2)$ のように，（正の値に）期待値を設定したベイズモデルもありうる．

2.3 平均消費電力のベイズ推定

さて，これらの事前分布と観測値 y_{ijk} から，平均消費電力 μ_{ij} の事後分布を求めることができる．これは通常のベイズ推定による．結果から書くと次のようになる．

$$\mu \sim N(S^{-1}\Sigma\bar{y}, S)$$

ここで μ は μ_{ij} を並べたベクトル，すなわち

$$\mu = (\mu_{11}, \mu_{21}, \mu_{31}, \ldots, \mu_{M1}, \mu_{12}, \mu_{22}, \ldots, \mu_{MN})^\top$$

である．選択肢 i の温度 t_j での測定回数を n_{ij} として，n は n_{ij} を並べたベクトルとすると，

$$\Sigma = \mathrm{diag}(\sigma^{-2}n)$$

である．また S は

$$S = \Sigma + \tau^{-2}I_{MN} - (\hat{\tau}_b^2\tau^{-4})T^\top T - K^T\hat{T}_aK$$

となる．ここでの I_{MN} は MN 次の単位行列，$\hat{\tau}_b^2$ は $t = (t_1, t_2, \ldots, t_N)^\top$ として

$$\frac{1}{\hat{\tau}_b^2} = \frac{1}{\tau_b^2} + \frac{Mt^\top t}{\tau^2}$$

で定義される定数である．さらに，T は長さ MN の横ベクトルで

$$T^\top = t \otimes 1_M$$

(ただし 1_M は 1 を M 個並べたベクトルで，$\otimes$ はクロネッカー積)，K は $M \times MN$ の行列で

$$K = \tau^{-2}J - \hat{\tau}_b^2\tau^{-4}N\bar{t}1_MT$$

(ただし $J = 1_N \otimes I_M$ で，$\bar{t} = N^{-1}1_N^\top t$ は t_j の平均値)，$\hat{T}_a$ は $M \times M$ の行列で

$$\hat{T}_a^{-1} = \left(\frac{1}{\tau_a^2} + \frac{N}{\tau^2}\right)I_M - \left(\frac{M^{-1}}{\tau_a^2} + \frac{\hat{\tau}_b^2N^2\bar{t}^2}{\tau^4}\right)1_M1_M^\top$$

である．

2.4 性能推定方式の導出

ここでは前節で示した推定式を導出する．以下の説明はベイズ推定の基本的な手法に従っているので，ベイズ推定に詳しい読者は読み飛ばしてよい．

ベイズ推定は，ベイズの定理

$$P(Y|X)P(X) = P(X,Y) = P(X|Y)P(Y)$$

に基づき，

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}$$

のように推定する．ここで X を未知パラメタ（ここでは μ_{ij}），Y を観測値（ここでは y_{ijk}）とする．$P(X)$ は X の**事前分布**，$P(X|Y)$ は X の**事後分布**と呼ばれる．事後分布は観測値 Y が与えられたときに，未知パラメタ X がどのような値でありそうかを確率分布で示したものとなる．なお，観測値 Y に対して $P(Y)$ は定数であるから，上の式は

$$P(X|Y) \propto P(Y|X)P(X)$$

とも書かれる．

本稿の μ_{ij} のモデルは未知パラメタ a_i, b を含んでおり，これらの未知パラメタに対してもベイズモデルが仮定されている．これが**階層ベイズ法**である．これらの a_i, b は**ハイパーパラメタ**と呼ばれるが，これらをまとめて Z とする．すなわち，X の事前分布は Z によって決まる $P(X|Z)$ の形をしていて，それに加えて Z に関する事前分布 $P(Z)$ も与えられている．これから X と Z に関する結合事前分布

$$P(X,Z) = P(X|Z)P(Z)$$

および X の周辺事前分布

$$P(X) = E_Z(P(X|Z)) = \int P(X|Z)P(Z)dZ$$

が決まる．これは $P(Z)$ に従って分布する Z の不確定性を取り込んだ結果得られる X の事前分布である．今回は $P(Y|X)$ は Z に依存しない仮定としているため，X の周辺事後分布は上の式を代入して

$$P(X|Y) \propto P(Y|X)P(X) = E_Z(P(Y|X)P(X|Z)) = \int P(Y|X)P(X|Z)P(Z)dZ$$

となる．

これを具体的に書き下すと

$$\begin{aligned}
P(Y|X) &= \prod_{ijk} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{(y_{ijk}-\mu_{ij})^2}{2\sigma^2}\right) \\
P(X|Z) &= \prod_{ij} \frac{1}{\sqrt{2\pi\tau^2}} \exp\left(\frac{(\mu_{ij}-a_i-bt_j)^2}{2\tau^2}\right) \\
P(Z) &= \frac{1}{\sqrt{2\pi\tau_b^2}} \exp\left(\frac{b^2}{2\tau_b^2}\right) \prod_i \frac{1}{\sqrt{2\pi\tau_a^2}} \exp\left(\frac{(a_i-a_0)^2}{2\tau_a^2}\right)
\end{aligned}$$

となる．これらを掛け合わせれば

$$P(Y|X)P(X|Z)P(Z) = C\exp Q$$

という形になる．ここで C は定数で，Q は

$$Q = \sum_{ijk} \frac{(y_{ijk}-\mu_{ij})^2}{2\sigma^2} + \sum_{ij} \frac{(\mu_{ij}-a_i-bt_j)^2}{2\tau^2} + \sum_i \frac{(a_i-a_0)^2}{2\tau_a^2} + \frac{b^2}{2\tau_b^2}$$

である．

さて，Q の右辺第 1 項には $\sum_k (y_{ijk}-\mu_{ij})^2$ という部分がある．これは

$$\begin{aligned}
\sum_k (y_{ijk}-\mu_{ij})^2 &= \sum_{ijk} y_{ijk}^2 - 2\mu_{ij}\sum_{ijk} y_{ijk} + n_{ij}\mu_{ij} \\
&= n_{ij}(\mu_{ij}-\bar{y}_{ij})^2 + \sum_{ijk} y_{ijk}^2 - n_{ij}\bar{y}_{ij}^2
\end{aligned}$$

と変形できる．ここで n_{ij} は (i,j) に対する k の個数，$\bar{y}_{ij} = n_{ij}^{-1}\sum_k y_{ijk}$ は (i,j) の標本平均である．これに着目して

$$\begin{aligned}
Q' &= \sum_{ij} \frac{(\mu_{ij}-\bar{y}_{ij})^2}{2\sigma^2/n_{ij}} + \sum_{ij} \frac{(\mu_{ij}-a_i-bt_j)^2}{2\tau^2} + \sum_i \frac{(a_i-a_0)^2}{2\tau_a^2} + \frac{b^2}{2\tau_b^2} \\
Q &= Q' + \sum_{ij} \frac{\sum_k y_{ijk}^2 - n_{ij}\bar{y}_{ij}^2}{2\sigma^2}
\end{aligned}$$

とする．ここで最後の式の右辺の第2項は y_{ijk} にのみ依存し，観測値が与えられれば定数となる．従って $\exp Q \propto \exp Q'$ であり，

$$P(Y|X)P(X|Z)P(Z) \propto \exp Q'$$

である．

次に a_0 に着目する．Q' のうち a_0 が関係する項を取り出して変形すると

$$\sum_i \frac{(a_i - a_0)^2}{2\tau_a^2} = \frac{\sum a_i^2 - 2Ma_0\bar{a} + Ma_0^2}{2\tau_a^2} = \frac{M(a_0 - \bar{a})^2}{2\tau_a^2} + \frac{\sum a_i^2 - M\bar{a}^2}{2\tau_a^2}$$

のようになる．ここで M は選択肢 i の数，$\bar{a}$ は a_i の平均値 $M^{-1}\sum_i a_i$ である．

$$Q'' = Q' - \frac{M(a_0 - \bar{a})^2}{2\tau_a^2}$$

とすると，Q'' は a_0 を含まない式となる．正規分布の式から

$$\int \frac{1}{\sqrt{2\pi\tau_a^2/M}} \exp\left(\frac{(a_0 - \bar{a})^2}{2\tau_a^2/M}\right) da_0 = 1$$

であることを我々は知っている．これから，

$$\int \exp Q' da_0 \propto \exp Q''$$

である．

同様に b に着目し，Q'' のうち b に関する項を取り出して変形すると，

$$\begin{aligned}
&\sum_{ij} \frac{(\mu_{ij} - a_i - bt_j)^2}{2\tau^2} + \frac{b^2}{2\tau_b^2} \\
&= b^2\left(\frac{1}{2\tau_b^2} + \frac{Mt^\top t}{2\tau^2}\right) - 2b\frac{1^T\sum t_j\mu_j - N\bar{t}1^Ta}{2\tau^2} + \frac{\sum_j(\mu_j - a)^\top(\mu_j - a)}{2\tau^2} \\
&= \frac{(b^2 - \hat{b})^2}{2\hat{\tau}_b^2} + \frac{\sum_j(\mu_j - a)^\top(\mu_j - a)}{2\tau^2} - \frac{\hat{b}^2}{2\hat{\tau}_b^2}
\end{aligned}$$

となる．ただし μ_j は μ_{ij} を並べたベクトル，a は a_i を並べたベクトルである．また，$\hat{\tau}_b^2$ は既出で，$\hat{b}$ は

$$\frac{1}{\hat{\tau}_b^2}\hat{b} = \frac{1^T\sum_j t_j\mu_j - N\bar{t}1^Ta}{\tau^2}$$

である．これより

$$Q^{(3)} = Q'' - \frac{(b^2 - \hat{b})^2}{2\hat{\tau}_b^2}$$

とすると，先と同様の議論により

$$\int \exp Q'' db \propto \exp Q^{(3)}$$

となる．

さらに同様に $Q^{(3)}$ のうち a に関する項を取り出して変形すると，若干の計算ののち

$$\frac{a^Ta - a^T11^Ta/M}{2\tau_a^2} + \frac{\sum_j(\mu_j - a)^T(\mu_j - a)}{2\tau^2} - \frac{\hat{\tau}_b^2(1^T\sum_j t_j\mu_j - N\bar{t}1^Ta)^2}{2\tau^4}$$
$$= \frac{(a-\hat{a})^\top\hat{T}_a^{-1}(a-\hat{a})}{2} - \frac{2a^\top K\mu + \frac{\mu^\top\mu}{2\tau^2}}{2} - \frac{\hat{\tau}_b^2\mu^\top T^\top T\mu}{2\tau^4}$$

となる．ここで多変数正規分布の式に従って a を積分すれば，

$$\begin{aligned} Q^{(4)} &= \frac{(\mu-\bar{y})^T\Sigma(\mu-\bar{y})}{2} + \frac{\mu^T\mu}{2\tau^2} - \frac{\hat{\tau}_b^2\mu^TT^T11^TT\mu}{2\tau^4} - \frac{\mu^TK^T\hat{T}_aK\mu}{2} \\ &= \frac{\mu^TS\mu}{2} - \frac{2\mu^T\Sigma\bar{y}}{2} + \text{定数} \end{aligned}$$

が残って，

$$P(X|Y) \propto \int P(Y|X)P(X|Z)P(Z)dZ \propto (\mu-\hat{\mu})^\top S(\mu-\hat{\mu})$$

となる．ただし $\hat{\mu} = S^{-1}\Sigma\bar{y}$ である．

3 おわりに

これまで著者は主に実験計画に関して研究を進めて，ワンステップ近似と呼ぶ一連のアルゴリズムを提案してきた．これは一定の成果を挙げてきており，その成果は自動チューニングのための数理コアライブラリ ATMathCoreLib に提供されている [5, 6]．この実験計画手法を最大限活用するためには，問題ごとのコストモデルを適切に定義することが必要である．そこで，現在はコストモデルの事例研究を進めている．本稿もその一つに位置付けられる．

本稿の第 1 節では，自動チューニングのというパラダイムの概要を，特に数理的な側面から論じた．また第 2 節では，消費電力の自動チューニングに向けて，温度の効果を考慮した消費電力モデルを構築した．このモデルでは，分散共分散行列 S も得られているので，消費電力の推定値と共に，その推定値がどのぐらい確からしいかについても情報が得られている．この 2 つ（推定値と，推定値の確からしさ）が得られれば，ワンステップ近似による自動チューニングが実現できる．

第 2 節で構築した消費電力モデルに基づき，ワンステップ近似を用いることで，消費電力や消費エネルギーを最小化するようなオンライン自動チューニングが可能になるものと期待される．今後，これを実装して，消費電力のオンライン自動チューニングを実現するソフトウェアを構築してゆきたい．

現時点では，オフライン自動チューニングはあまりうまくゆかないと考えている．温度という制御の困難な特徴量があるため，基礎消費電力 a_i を十分な精度で評価できないからである．ただし温度依存性 b が選択肢によらない定数であることを利用すれば，一定の条件下でオフライン自動チューニングができる可能性もある．この方向性での可能性も検討したい．

謝辞

本研究の一部は，JST CREST 「ULP-HPC: 次世代テクノロジのモデル化・最適化による超低消費電力ハイパフォーマンスコンピューティング」および文科省科研費基盤 (A) 「汎用自動チューニング機構を実現するためのソフトウェア基盤の研究」による．

参考文献

[1] R. Suda, "A Bayesian Method for Online Code Selection: Toward Efficient and Robust Methods of Automatic Tuning," Second International Workshop on Automatic Performance Tuning (iWAPT2007), Sep. 2007, pp. 23–31.

[2] T. Katagiri, K. Kise, H. Honda and T. Yuba, ABCLibScript: A directive to support specification of an auto-tuning facility for numerical software, *Parallel Computing* **32** (2006), 92-112.

[3] R. Suda, "Methods of Parallel Experimental Design of Online Automatic Tuning and their Application to Parallel Sparse Matrix Data Structure," Proc. Fifth international Workshop on Automatic Performance Tuning (iWAPT 2010).

[4] 須田礼仁,「自動チューニングにおける選択肢絞り込み」, 日本応用数理学会 2012 年度年会, 予稿集 271-272, 2012 年 8 月.

[5] 須田礼仁, 「自動チューニング数理基盤ライブラリ ATMathCoreLib」, 情報処理学会 研究報告 HPC-129-14, 2011 年 3 月.

[6] ATMathCoreLib: http://olab.is.s.u-tokyo.ac.jp/~reiji/atmathcorelib/